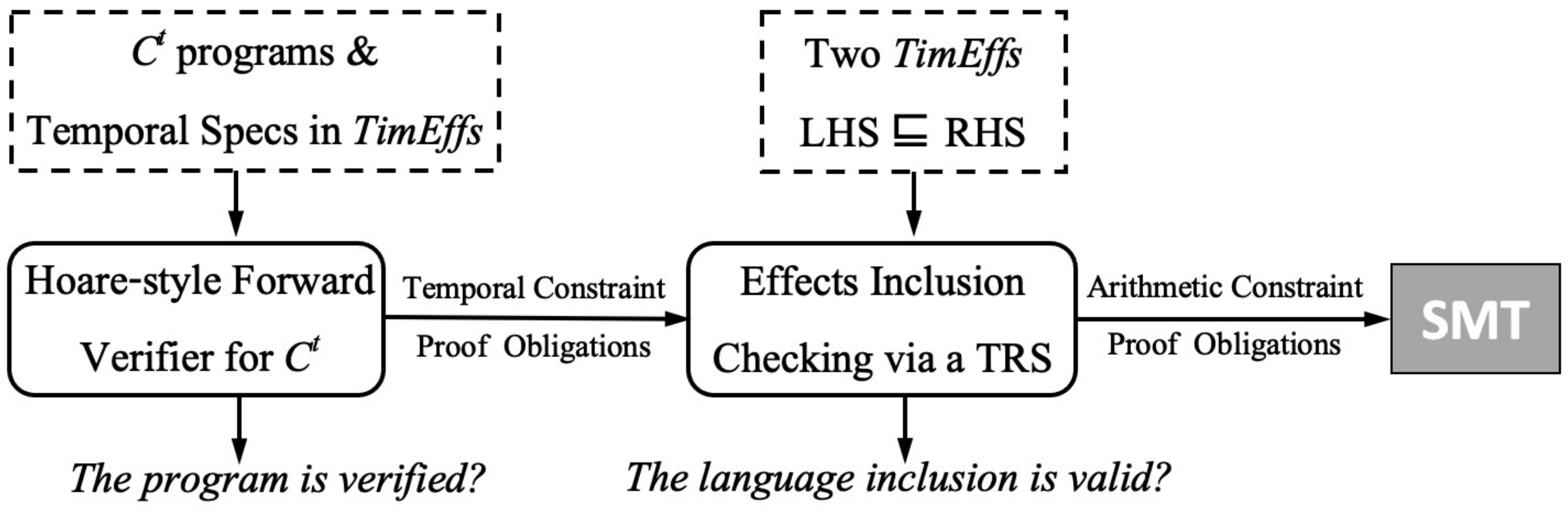
Automated Temporal Verification for Real-Time Systems via Implicit Clocks and an Extended Antimirov Algorithm

Yahui Song @ SPLASH-SRC 2022 ✉ yahuis@comp.nus.edu.sg

National University of Singapore

Overview

To go beyond the existing *Timed Automata (TA)* based techniques, we propose a novel solution that integrates a modular Hoare-style forward verifier with a term rewriting system (TRS) on Timed Effects (*TimEffs*). The main purposes are to: increase the expressiveness, dynamically manipulate clocks, and efficiently solve clock constraints. Contributions are:



C^t programs & Temporal Specs in *TimEffs*

Two *TimEffs* LHS ⊑ RHS

Hoare-style Forward Verifier for C^t

Temporal Constraint Proof Obligations

Effects Inclusion Checking via a TRS

Arithmetic Constraint Proof Obligations

SMT

The program is verified?

The language inclusion is valid?

- Language Abstraction, C^t: generalizes the real-time systems with mutable variables and timed behavioural patterns, e.g., delay, timeout, deadline.
- Novel Specification, *TimEffs*: extends regular expressions with dependent values and arithmetic constraints.
- Efficient Term Rewriting System, *TRS*: solves inclusions between *TimEffs*, by iterated checking of their *derivatives*.

TimEffs (Symbolic Timed Automata)

```
void addOneSugar()
/* req: true ∧ _*
  ens: t>1 ∧ ϵ # t */
{ timeout ((), 1); }

void addNSugar (int n)
/* req: true ∧ _*
  ens: t≥n ∧ EndSugar#t */
{ if (n == 0)
    event["EndSugar"];
  else {
    addOneSugar();
    addNSugar (n-1);}}
```

```
void makeCoffee (int n)
/* req: n≥0 ∧ _*· CupReady
  ens: n≤t≤5 ∧ t'≤4 ∧ */
    (EndSugar # t) · (Coffee # t') */
{ deadline (addNSugar(n), 5);
  deadline (event["Coffee"],4)}

int main ()
/* req: true ∧ ϵ
  ens: t≤9 ∧ ((!Done)* # t) · Done*/
{ event["CupReady"];
  makeCoffee (3);
  event["Done"];}
```

$$(\textit{Timed Effects})\ \Phi ::= \pi \wedge \theta \mid \Phi_1 \vee \Phi_2$$

$$(\textit{Event Sequences})\ \theta ::= \bot \mid \epsilon \mid ev \mid \theta_1 \cdot \theta_2 \mid \theta_1 \vee \theta_2 \mid \boxed{\theta_1 || \theta_2 \mid \pi?\theta \mid \theta\#t} \mid \theta^{\star}$$

$$(\textit{Events})\ ev ::= \mathtt{A}(v, \alpha^{*}) \mid \tau(\pi) \mid \overline{\mathtt{A}} \mid \_$$

$$(\textit{Pure})\ \pi ::= True \mid False \mid bop(t_1, t_2) \mid \pi_1 \wedge \pi_2 \mid \pi_1 \vee \pi_2 \mid \neg\pi \mid \pi_1 \Rightarrow \pi_2$$

$$(\textit{Real-Time Terms})\ t ::= c \mid x \mid t_1 + t_2 \mid t_1 - t_2$$

$c \in \mathbb{Z}$ $x \in \mathbf{var}$ $(\textit{Real Time Bound})\ \#$ $(\textit{Kleene Star})\ \star$

Our proposal overcomes the following existing limitations:

1) TAs cannot be used to specify/verify incompletely specified systems (i.e., whose timing constants have yet to be known) and hence cannot be used in early design phases;
2) verifying a system with a set of timing constants usually requires enumerating all of them if they are integer-valued;
3) TAs cannot be used to verify systems with timing constants to be taken in a real-valued dense interval.

Language Inclusion – the Antimirov Algorithm

Our TRS is an extension of Antimirov and Mosses's algorithm, which can be deployed to decide inclusions of two regular expressions (REs) through an iterated process of checking inclusions of their partial derivatives.

Definition 1 (Derivatives). *Given any formal language S over an alphabet Σ and any string u∈ Σ*, the derivatives of S w.r.t u is defined as:* $u^{-1}S = \{w \in \Sigma^{*} \mid uw \in S\}$.

Definition 2 (Regular Expression Inclusion). *For REs r and s,*

$$r \preceq s \Leftrightarrow \forall A (A \in \Sigma).\ A^{-1}(r) \preceq A^{-1}(s).$$

Definition 3 (*TimEffs* Inclusion). *For TimEffs* Φ_1 *and* Φ_2,

$$\Phi_1 \sqsubseteq \Phi_2 \Leftrightarrow \forall A.\ \forall t \geq 0.\ (A\#t)^{-1}\Phi_1 \sqsubseteq (A\#t)^{-1}\Phi_2.$$

➢ Antimirov V M, Mosses P D. Rewriting extended regular expressions[J]. Theoretical Computer Science, 1995, 143(1): 51-72.

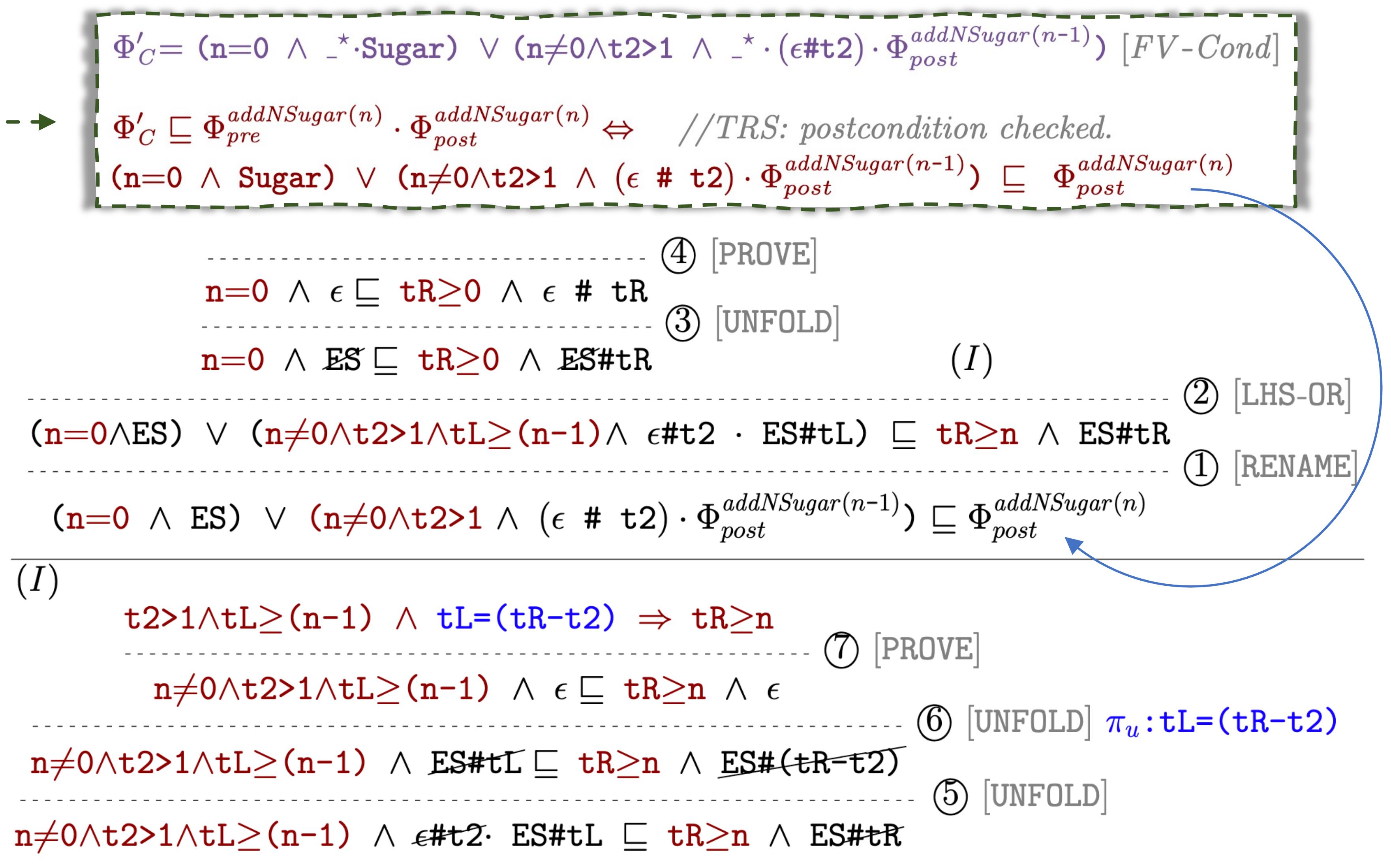
Expressiveness of *TimEffs*

TimEffs draw similarities to Metric Temporal Logic (MTL), derived from LTL, where a set of non-negative real numbers is added to temporal modal operators. Basic operators are: □ for "globally"; ◇for "finally; ○ for "next"; U for "until", and their past time reversed versions: $\overleftarrow{\Box}$; $\overleftarrow{\Diamond}$; and ⊖ for "previous"; S for "since".

Φ_{post}	$\Box_I \mathtt{A} \equiv (\mathtt{A}^{\star})\#t$	$\Diamond_I \mathtt{A} \equiv (\_^{\star} \cdot \mathtt{A})\#t$	$\bigcirc_I \mathtt{A} \equiv (\_)\#t \cdot \mathtt{A}$	$\mathtt{A}\mathcal{U}_I\mathtt{B} \equiv (\mathtt{A}^{\star})\#t \cdot \mathtt{B}$
Φ_{pre}	$\overleftarrow{\Box}_I \mathtt{A} \equiv (\mathtt{A}^{\star})\#t$	$\overleftarrow{\Diamond}_I \mathtt{A} \equiv (\mathtt{A} \cdot \_^{\star})\#t$	$\ominus_I \mathtt{A} \equiv \mathtt{A} \cdot ((\_)\#t)$	$\mathtt{A}\mathcal{S}_I\mathtt{B} \equiv \mathtt{B} \cdot ((\mathtt{A}^{\star})\#t)$

TimEffs in the precondition, encode past-time temporal specifications. I in MTL is the time interval with concrete upper/lower bounds; whereas in TimEffs they can be symbolic bounds, dependent on program inputs.

A Demonstration of the Automated TRS



$$\Phi'_C = (n=0 \wedge \_^{\star}\cdot Sugar) \vee (n\neq0 \wedge t2>1 \wedge \_^{\star}\cdot(\epsilon\#t2)\cdot\Phi_{post}^{addNSugar(n-1)})\ [FV\text{-}Cond]$$

$$\Phi'_C \sqsubseteq \Phi_{pre}^{addNSugar(n)} \cdot \Phi_{post}^{addNSugar(n)} \Leftrightarrow \quad //TRS:\ postcondition\ checked.$$

$$(n=0 \wedge Sugar) \vee (n\neq0 \wedge t2>1 \wedge (\epsilon\ \#\ t2)\cdot\Phi_{post}^{addNSugar(n-1)}) \sqsubseteq \Phi_{post}^{addNSugar(n)}$$

④ [PROVE]

$$n=0 \wedge \epsilon \sqsubseteq tR\geq0 \wedge \epsilon\ \#\ tR$$

③ [UNFOLD]

$$n=0 \wedge \cancel{ES} \sqsubseteq tR\geq0 \wedge \cancel{ES}\#tR \qquad (I)$$

② [LHS-OR]

$$(n=0 \wedge ES) \vee (n\neq0 \wedge t2>1 \wedge tL\geq(n-1) \wedge \epsilon\#t2 \cdot ES\#tL) \sqsubseteq tR\geq n \wedge ES\#tR$$

① [RENAME]

$$(n=0 \wedge ES) \vee (n\neq0 \wedge t2>1 \wedge (\epsilon\ \#\ t2)\cdot\Phi_{post}^{addNSugar(n-1)}) \sqsubseteq \Phi_{post}^{addNSugar(n)}$$

(I)

$$t2>1 \wedge tL\geq(n-1) \wedge tL=(tR-t2) \Rightarrow tR\geq n$$

⑦ [PROVE]

$$n\neq0 \wedge t2>1 \wedge tL\geq(n-1) \wedge \epsilon \sqsubseteq tR\geq n \wedge \epsilon$$

⑥ [UNFOLD] $\pi_u: tL=(tR-t2)$

$$n\neq0 \wedge t2>1 \wedge tL\geq(n-1) \wedge \cancel{ES\#tL} \sqsubseteq tR\geq n \wedge \cancel{ES\#(tR-t2)}$$

⑤ [UNFOLD]

$$n\neq0 \wedge t2>1 \wedge tL\geq(n-1) \wedge \cancel{\epsilon\#t2\cdot}\ ES\#tL \sqsubseteq tR\geq n \wedge ES\#\cancel{tR}$$

Limitations of Our TRS.

Our TRS is incomplete, meaning there exist valid inclusions which will be disproved in our system. That is mainly because of insufficient unification in favour of achieving automation.

Scan Me For the Project Repository